IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

AF/2124 IFW

Group Art Unit: 2124
Examiner: W.H. Wood

In re Application of:	Hartnett et al.
Title:	System and Method for Controlling the Entry of Instructions into a Pipeline of an Instruction Processor
Serial No.:	09/419,439
Filed:	October 15, 1999
Docket No.:	RA 5274
Customer No.	27516

Date: July 1, 2004

Commissioner of Patents
MS Appeal Brief - Patents
P O Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF TRANSMITTAL

Sir:

Transmitted herewith is an Appeal Brief for this application. Applicant is other than a small entity.

We are transmitting herewith the attached:

- Appellant's Appeal Brief Filed Under 37 C.F.R. 1.193 (d) in Triplicate.

FEE PAYMENT

Please charge Account No. 19-3790 the sum of $330.00. It is believed no extension of time is required under 37 C. F. R. 1.136(a). However, if any extension of time or any additional fee is required, please charge Account No. 19-3790.

A duplicate of this transmittal is attached.

Respectfully submitted,

Beth L. McMahon

Beth L. McMahon
Reg. No.: 41,987
Tel. No.: (651) 635-7893
Unisys Corporation
M.S. 4773
P.O. Box 64942
St. Paul, MN 55164-0942

I hereby certify that this correspondence is being deposited in the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450 on July 1, 2004.

Beth L. McMahon
Attorney for Applicants

Beth L. McMahon
Signature

July 1, 2004
Date of Signature

APPEAL TO THE BOARD OF PATENT APPEALS AND INTERFERENCES

Art Group Unit No. 2124
Examiner: W. H. Wood

July 1, 2004

Customer Assignment No.	27516
Serial No.:	09/419,439
Filed:	October 15, 1999
In re Application of:	Hartnett et al.
Title:	SYSTEM AND METHOD FOR CONTROLLING THE ENTRY OF INSTRUCTIONS INTO A PIPELINE OF AN INSTRUCTION PROCESSOR
Attorney Docket No.	RA-5274

APPELLANT'S BRIEF

MS Appeal Brief - Patents
Commissioner of Patents
P.O. Box 1450
Alexandria, VA 22313-1450

This brief is being submitted within two (2) months of receipt of the Notice of Appeal on May 3, 2004, by the U.S. Patent and Trademark Office. This brief, which is filed in triplicate, is transmitted with the required fee. Permission is hereby granted to charge deposit account number 19-3790 for any errors in fee calculation. Appellants request this Appeal Brief be made of record and fully considered.

CERTIFICATE OF MAILING (37 CFR 1.8(a))

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail, in an envelope addressed to: Box MS Appeal Brief - Patents, Commissioner of Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date shown below:

Beth McMahon — July 1, 2004
(Beth McMahon — *(Date)*

TABLE OF CONTENTS

Real Party In Interest

The real party in interest is Unisys Corporation, with an address as follows:

Unisys Corporation
Township Line and Union Meeting Roads
Blue Bell, Pennsylvania 19424

Unisys Corporation is the real party in interest through an assignment from all of the inventors of their entire interest, recorded on 10/15/1999 in the USPTO at Reel/Frame 010322/0862.

Related Pending Appeals or Interferences

There are no pending appeals or interferences related to the subject Appeal.

Status of Claims

Claims 1-22 remain pending in the application as follows:

Claims 1, 2, 5, 6, 11-14, 19 and 20 stand finally rejected; and

Claims 3, 4, 7-10, 15-18, 21 and 22 are objected to as depending from a rejected base Claim.

Status of the Amendments

A first amendment was submitted on August 2, 2002 to amend Claims 1, 11-15, and 18. This amendment has been entered by the Examiner. No further

amendments to the Claims were submitted or entered. A clean copy of Claims 1-22, as amended, is provided as Appendix A.

Summary of the Invention

Applicants' invention provides a system and method for controlling the instruction pipeline of a data processing system. As is known in the art, an instruction pipeline is a circuit that is capable of executing multiple instructions at once in an "assembly line" type fashion. A pipeline includes multiple stages, each of which performs a portion of the instruction execution steps. When operating in a fully overlapped manner, a different instruction is executing within each of the pipeline stages.

In one embodiment of Applicants' system, an exemplary pipeline is described that includes six execution stages. A simple type of instruction called a "single-cycle" instruction is capable of moving through this pipeline in six cycles of the system clock, spending one clock cycle in each of the six stages. If all instructions entering the pipeline are single-cycle instructions, the pipeline is capable of beginning execution on six instructions within six clock cycles.[1]

According to the system and method of Applicants' invention, the number of instructions that enter the pipeline within six clock cycles may be throttled to something less than six instructions. A user is allowed to programmably specify a count value M that indicates the number of instructions that is to begin

[1] Applicants' Specification ("Specification"), page 13 line 5 - page 14 line 15 in reference to Figures 1 and 2.

execution during six clock cycles.[2] A logic sequencer that is coupled responsively to the pipeline receives this count value[3]. After being enabled, the sequencer controls entry of instructions into the pipeline so that during the six clock cycles that follow, no more than M instructions enter the pipeline. [4] For example, if M is programmed to a value of "five", the sequencer exerts control so that, at most, five instructions enter the pipeline to begin concurrent execution during the following six clock cycles. This type of operation is shown in Applicants' Figure 14. [5] The value M may be selected as any positive integer less than six, as shown in Applicants' Figures 10 through 13.

It may be noted that while the specific embodiment of the invention discussed above is capable of controlling the pipeline so that M instructions begin execution within six clock cycles, the time period of "six clock cycles" is largely arbitrary. Applicants' pipeline control mechanism may be readily adapted for use with pipelines of other sizes. In these alternative embodiments, the time period during which Applicants' sequencer exerts control over a pipeline may be something other than six clock cycles. Thus, Applicants' invention may be generalized as a system and method for controlling the number of instructions entering the pipeline to begin concurrent execution within *a predetermined period of time*, wherein this predetermined time period may be six clock cycles, or some other known time period that is dependent upon the implementation.

---

[2] Specification page 7 lines 15-20.
[3] See Specification page 25 line 5 - page 26 line 23 in reference to Figure 9.
[4] Specification, page 7 lines 15 - 20.
[5] See, also, Specification page 33 lines 1 - 4 in reference to Figure

Applicants' invention finds a number of uses. First, it may be employed to prevent various types of conflicts that exist between both contiguous and non-contiguous instructions in the instruction stream. For example, consider an instruction stream containing a sequence of instructions N, N+1, and N+2. Instruction N may generate a value used during execution of instruction N+2. If this generated value is not available when execution of instruction N+2 requires its use, a conflict error arises. Applicants' invention may be used to control entry of instructions N+1 and/or N+2 into the pipeline so that this type of conflict is avoided.[6] Applicants' invention may also be employed to match the throughput of a data processing system to a peripheral device.[7]

The functionality provided by Applicants' invention cannot be fully appreciated without considering the complexity of instruction pipelines. A typical instruction set includes multiple types of instructions. As discussed above, single-cycle instructions generally spend one clock cycle in each stage of the pipeline. Other instructions require more than a single cycle to proceed from one stage of the pipeline to the next. For example, during execution of an "extended-cycle" instruction, one or more additional clock cycles are required before the instruction can proceed from the first to the second execution stage. The number of additional clock cycles that is required is specific the instruction.[8] During these additional cycles, the next instruction in the instruction stream is

---

14.

[6] Specification, page 7 line 27 - page 8 line 4.

[7] Specification, page 23 line 4 - page 25 line 3.

[8] Specification, page 17 lines 7 - 24 in reference to Figure 6.

not allowed to enter the pipeline. For this reason, the manner in which instructions enter the pipeline cannot be predicted without knowing the specific instruction sequence.

Another source of unpredictability in the operation of instruction pipelines involves the availability of instructions and operands. In many pipelines, an attempt is made to retrieve both instructions and any required operands from cache memory prior to, or during, instruction execution. If a cache miss occurs, however, the retrieval of an instruction or operand is likely delayed.[9] This may delay entry of additional instructions into the pipeline. The occurrence and size of this delay cannot be readily predicted prior to execution of the instruction stream.

Still another example of the unpredictability associated with pipelined execution involves the occurrence of errors and interrupts. When pipeline logic detects these types of events, instructions within some stages of the pipeline are prevented from advancing within the pipeline, whereas execution is allowed to complete within other stages of the pipeline.[10] Again, this may cause delay in the entry of instructions into the pipeline.

To summarize, many variables affect the way in which instructions enter instruction pipelines. The sequencer of Applicants' invention must take all of these unpredictable occurrences into account to control the number of instructions that enter the pipeline during a predetermined period of time. For

---

[9] Specification, page 16 lines 16 - 22 in reference to Figure 5.
[10] Specification, page 26 lines 3 - 16.

example, assume a count value has been selected to cause three instructions to enter the pipeline during the predetermined period of time, which in one embodiment is six clock cycles. Assume, further, that during the six clock cycles in question, an extended-cycle instruction is executing that delays entry of subsequent instructions into the pipeline. Applicants' sequencer must take this naturally occurring delay into account when controlling the entry of instructions into the pipeline so that the specified number of instructions begins concurrent execution within the pipeline in the predetermined period of time.

According to one aspect of Applicants' invention, Applicants' sequencer may be programmed to operate in any of multiple modes. In one mode, Applicants' sequencer uses the specified count value to throttle entry of instructions into the pipeline during each successive six-cycle time period. In other modes of operation, a count value is associated with a particular instruction or instruction combination. When the sequencer detects the instruction or instruction combination, it is enabled to limit the number of instructions entering the pipeline during the next six clock cycles to that number specified by the associated count value.[11] In still another mode, the sequencer is enabled to throttle entry of instructions into the pipeline by the detection of a selected combination of instructions in conjunction with the occurrence of a system condition such as an error or interrupt.[12]

---

[11] Specification, page 9 line 14 - page 9 line 8.
[12] Specification, page 9 line 25 - page 10 line 7.

Aspects of Applicants' invention discussed above are summarized in Applicants' representative Claim 1 as follows:

> "For use in a data processing system having ...an instruction pipeline capable of initiating simultaneous execution on a variable number of the instructions in a predetermined period of time, a system for programmably controlling the variable number of the instructions..., comprising:
>
> a first storage device to receive and to store a programmable count value indicative of a predetermined number of instructions; and
>
> a logic sequencer ... to receive said programmable count value, and ...to cause the instruction pipeline to receive, and to initiate concurrent execution on, the predetermined number of the instructions in the predetermined period of time." (Claim 1 lines 1, 3-13.)

Issues

I. Whether Claims 1, 2, 5, 11-12 and 19-20 are unpatentable under 35 USC §103(a) over U.S. Patent Number 6,029,006 to Alexander et al. ("Alexander").

II. Whether Claims 6 and 13-14 are unpatentable under 35 USC §103(a) over Alexander in view of U.S. Patent Number 5,996,064 to Zaidi et al. ("Zaidi") and further in view of U.S. Patent Number 6,345,362 to Bertin et al. ("Bertin").

Grouping of the Claims

Claims 1, 2, 5, 11-12 and 19-20 stand or fall together; and

Claims 6 and 13-14 stand or fall together.

Arguments

I. Whether Claims 1, 2, 5, 11-12 and 19-20 are unpatentable under 35 USC §103(a) over U.S. Patent Number 6,029,006 to Alexander et al. ("Alexander").

Before discussing the first issue in detail, the pertinent aspects of Applicants' representative Claim 1 are summarized for discussion purposes. As discussed above, Applicants' Claim 1 describes a data processing system having an instruction pipeline capable of initiating simultaneous execution on a specified number of instructions in a predetermined period of time. The system includes a first storage device to store a programmable count value that is indicative of a predetermined number of instructions. The system further includes a logic sequencer to cause the instruction pipeline to initiate concurrent execution on the predetermined number of instructions within the predetermined period of time. In one exemplary embodiment, the predetermined period of time is six clock cycles, although other time periods may be used in alternative embodiments. The remaining independent Claims 11 and 19 include aspects that are similar to those included within Claim 1.

Next, the system of Alexander is considered for discussion purposes. Alexander teaches a data processing system that incorporates a throttling circuit

for limiting power consumption.[13] The throttling circuit includes a register that stores an interval field. This field specifies an interval by which the fetching of each instruction in the instruction stream is delayed.[14] After an instruction is fetched, it enters the pipeline of the system for execution. Thus, the Alexander interval field is used to determine the delay between the entry into the pipeline of two successive instructions in the instruction stream.

Operation of the Alexander system may be illustrated by the following example. Assume that the Alexander interval is set to "one". According to this configuration, three successive instructions X, Y, and Z will enter the Alexander five-stage pipeline with a one-cycle delay between instructions X and Y, and another cycle delay between instructions Y and Z, as follows:

Stage:	**1**	**2**	**3**	**4**	**5**
Clock Cycle					
Cycle 1	X				
Cycle 2		X			
Cycle 3	Y		X		
Cycle 4		Y		X	
Cycle 5	Z		Y		X

The Alexander system may next be considered in reference to Applicants' Claim 1. The Examiner states that the Alexander interval field suggests

[13] Alexander Abstract line 1.
[14] Alexander column 2 lines 54 - 56.

Applicants' count value of Claim 1. As previously discussed, Applicants' count value indicates the number of instructions that will be allowed to enter an instruction pipeline to begin concurrent execution within a predetermined period of time. Consider, for example, an embodiment of Applicants' invention wherein the predetermined period of time is "five clock cycles".[15] In this embodiment, a count value may be selected to control the number of instructions that will begin concurrent execution during the five clock cycles following enabling of the system. For example, by selecting a count value of "four", four instructions are allowed to begin execution during the following five clock cycles, and so on. This precise control is provided by a complex set of control sequences described within Applicants' Specification and illustrated within the various timing and logic diagrams.[16] This complex mechanism allows Applicants' system to "de-pipe", or partially clear, the pipeline in the most efficient manner possible to handle specific timing conflicts.

The type of precise pipeline control provided by Applicants' system cannot be achieved using the Alexander mechanism. For example, in the Alexander system, there is no interval value that can be selected to cause exactly four instructions to enter the pipeline in five clock cycles. This is apparent from the scenario illustrated above. Moreover, in Alexander, there is

---

[15] In Applicants' exemplary embodiment, the predetermined period of time is "six clock cycles", which corresponds to a six-stage pipeline. However, the invention may be adapted for any N-stage pipeline, as discussed on page 7, lines 7-14, of the Specification.

[16] See Applicants' Figures 11-14, which provide examples of selecting count values of 2-5 instructions, respectively, in a six stage pipeline.

no easy way to determine how many instructions will enter the Alexander pipeline as the result of the selection of any given interval value. For example, there is no clear correlation between selection of an Alexander interval value of "three" or "four" and the number of instructions that will enter the Alexander instruction pipeline within a given period of time. In fact, the number of instructions that will enter the pipeline in Alexander during a predetermined time period will vary no matter which delay value is specified. This is because the Alexander control system inserts the same delay between instructions regardless of whether an instruction is a single-cycle instruction, or a complex instruction requiring more than a single clock cycle per stage to execute.[17]

The pipeline control provided by Applicants' claimed system affords certain advantages not offered by the Alexander system. For example, it has been found that merely inserting a particular delay between successive instructions in the instruction stream will not address certain timing conflicts associated with non-contiguous instructions or certain instruction combinations. This is discussed in Applicants' Specification, as follows:

> "A system for controlling pipeline execution in a programmable manner is described in U.S. Patent No. 5,911,083.... This patent describes a system for preventing additional instructions from entering the instruction pipeline for a selected amount of time after selected ones of the instructions enter the instruction pipeline..... [This prior art] de-piping mechanism *cannot be used* to efficiently solve timing conflicts that are caused by two non-contiguous instructions within the instruction stream, or that are caused by a combination of more than two instructions. This is because the prior art de-piping mechanism inserts delay into the pipeline

[17] Alexander column 5 lines 28 - 35.

immediately following a particular instruction instead of controlling the number of instructions that are concurrently executing within the pipeline."[18]

As is evident from the above discussion, Applicants' system for controlling the *number of instructions entering a pipeline within a given period of time* provides more precise pipeline control than does the Alexander system for controlling *delay* inserted between two successive instructions entering a pipeline. The precise control offered by Applicants' system is not trivial: it requires a complex control mechanism that takes into account the variables that affect the way in which instructions move through the pipeline.

As discussed above in the Summary of the Invention, instructions do not always advance between the stages of a pipeline each clock cycle, nor do instructions necessarily advance between pipeline stages in lock step with one another. This can be caused by the execution of extended-cycle instructions or the occurrence of interrupts, errors, and cache misses. These types of occurrences may delay entry of some instructions into the pipeline, and/or may cause some stages of instruction execution to be extended. When any of these events occur, the number of instructions entering the pipeline in a given period of time may be affected. Applicants' system and method take these events into account when controlling the number of instructions entering a pipeline within a given period of time.[19]

---

[18] Specification page 5 lines 4-11 and 24-28, emphasis added.

[19] See, for example, Applicants' Specification page 33, lines 13-25 in reference to use of Applicants' invention when an extended-cycle instruction is

The foregoing can best be appreciated by example. Assume that Applicants' sequencer is programmed to control entry of four instructions into the pipeline within six clock cycles. After the sequencer is so enabled, an extended-cycle instruction enters the first stage of the pipeline. This instruction requires one extra clock cycle to complete the first stage of execution. This naturally delays the entry of the next instruction into the pipeline by one clock cycle. During the next four clock cycles, Applicants' system will take this natural delay into account when controlling entry into the pipeline of the next three instructions.[20] This produces the required throughput of four instructions within six clock cycles. In contrast, the Alexander system inserts delay between each instruction in the instruction stream regardless of whether any naturally occurring delay may have already been inserted into the instruction stream. This unnecessarily slows throughput, and in some cases, may not resolve conflicts associated with non-contiguous instructions.[21]

The above-described considerations also apply to other situations wherein it is desirable to precisely control the entry of instructions into the pipeline. For example, it may be desirable to precisely match processing throughput of a pipeline to a peripheral device[22]. It is not possible to provide the necessarily level of control using a system such as taught by Alexander,

---

executing.

[20] See Applicants' Figure 16, and Applicants' Specification page 33 line 26 - page 34 line 22.

[21] Specification page 5 lines 24-28, discussing a prior art system that adds delay between two instructions.

[22] Specification page 8 lines 10 - 13.

which is designed to exert inexact instruction throttling adequate for limiting power consumption, but which is not suitable for fine-tuning pipeline operation.

To summarize, Applicants' claimed invention utilizes a very different, and more complex, control mechanism than that described in Alexander. Using this mechanism, Applicants' system is capable of controlling the number of instructions entering a pipeline in a given period of time. This system provides important benefits that cannot be realized by the Alexander mechanism. For at least these reasons, Applicants' invention of Claims 1, 2, 5, 11-12 and 19-20 is patentable over Alexander.

Next, it may be considered whether a prima facie case of obviousness has been established. Three criteria have been established for setting forth a prima facie case of obvious.[23] First, there must be some suggestion or motivation, either in the references themselves, or in the knowledge available to one skilled in the art, to modify a reference. Second, there must be a reasonable expectation of success. Third, the prior art references must teach or suggest all claim limitations.

In regards to the first element of the prima facie case, there is no motivation, either in Alexander itself, or anywhere in the art, to modify the Alexander delay insertion mechanism to obtain Applicants' system. As stated previously, the Alexander system is designed to limit power consumption and reduce heat dissipation. A measurable change in heat dissipation will be achieved by changing system throughput by hundreds, if not thousands, of

instructions per second. This level of control can be adequately accomplished using the simple delay mechanism of Alexander, which includes a control register, a counter, and minimal interconnecting logic[24]. Modifying this logic to include anything more complex would be entirely unnecessary and even counter productive. A more complex circuit of the type required to provide Applicants' pipeline control mechanism would only *increase* power consumption, and would be at odds with the very purpose of the Alexander system. Thus, there is absolutely no motivation to modify Alexander in a way that would teach or suggest Applicants' system and method.

When discussing motivation, the Examiner maintains that it would have been obvious to implement Alexander's processor using a synchronous pipeline such as Applicants' because one of ordinary skill in the art would be motivated to provide the benefits of power conservation to all types of processors.[25] Applicants disagree with this statement. Among those skilled in the art, it has long been recognized that minimizing power consumption and maximizing performance are generally competing interests. This is illustrated by the Alexander system, which reduces power consumption by limiting processor performance. In contrast to the Alexander system, a pipeline of the type described in Applicants' system is designed to execute with maximum efficiency without particular regard to power consumption. To this end, Applicants' invention provides a system and method capable of resolving specific timing

---

[23] Manual of Patent Examining Procedure, §2143

[24] See Alexander Figure 17.

conflicts without unnecessarily slowing processor throughput.[26] This purpose would be defeated by use of a system such as taught by Alexander. Moreover, as discussed in the foregoing paragraph, even if one skilled in the art would be motivated to provide the benefits of power conservation to all types of processors, it is not understood how such motivation would result in modification of Alexander in any way that would teach or suggest Applicants' system.

Further in regards to motivation, the Examiner asserts that "...one of ordinary skill in the art at the time of invention would understand the processor itself defines a predetermined period of time (synchronous pipeline) and the interval of Alexander defines the variable number of instructions that are being executed in the pipeline (by allowing only so many instructions into the defined period of time of the pipeline)." [27] While this statement is not completely understood, the Examiner appears to be asserting that the Alexander delay interval defines the *number* of instructions being executed within the pipeline. As previously discussed, this most certainly is not the case. The Alexander delay interval defines just that: *a delay*.

While a delay such as taught by Alexander does have an indirect effect on instruction throughput, selection of any given delay cannot control the number of instructions entering the pipeline in a given period of time. Although lengthening the delay will result in somewhat fewer instructions entering the pipeline over time, and shorting the delay will have the opposite effect, no

---

[25] Final Rejection ("Final Rejection") dated 1/29/2004, page 3, lines 6-10.
[26] Specification page 9 lines 5 - 8.

particular delay can be selected to actually exert control over the number of instructions executing within a given period of time. This is because mere insertion of delay between successive instructions is not precise enough, and does not take into account unpredictable occurrences within the pipeline.

Turning next to the second element of a prima facie case of obviousness, there must be a reasonable expectation of success when modifying a reference to obtain the claimed invention. As previously discussed, the Alexander system regulates instruction processing using a simple delay mechanism that is suitable for the intended purpose of reducing power consumption. Modifying this logic to teach or suggest Applicants' claimed invention would require the addition of complex sequencer logic. This would *increase* power consumption, and is at odds with the very purpose of the Alexander system. Conversely, incorporating the Alexander delay mechanism into a high-speed pipeline such as that of Applicants' system would defeat the goals of speed and efficiency. From either standpoint, there is no reasonable expectation that modifying Alexander to teach or suggest Applicants' system would result in success.

Finally, the third element of a prima facie case of obviousness is discussed. According to this element, the prior art reference must teach or suggest all claim limitations. As set forth above, Applicants' representative Claim 1 includes:

a.) a device to store a count value indicative of a predetermined number of instructions; and

---

[27] Final Rejection, page 3 lines 10 - 15.

b.) a sequencer to cause the pipeline to receive and initiate concurrent execution on the predetermined number of instructions in the predetermined period of time.

With respect to the elements of Claim 1, Alexander does not teach, or even suggest, any sort of a count for indicating a predetermined number of instructions. Additionally, Alexander does not teach, or even suggest, any type of sequencer that can cause the number of instructions, as indicated by some specified count value, to begin concurrent execution within the pipeline in any predetermined period of time. The Alexander interval value *does not* teach or suggest this count value, and the Alexander delay circuit[28] does not, and could not, control the pipeline so a specified number of instructions enters the pipeline in a predetermined amount of time.

To summarize, Applicants' invention of representative Claim 1 provides a very different control mechanism as compared to that described in Alexander. This control mechanism allows a pipeline to be controlled in a precise manner that cannot be achieved from the system of Alexander. For at least these reasons, Claims 1, 2, 5, 11-12, and 19-20 are patentable over Alexander. In addition, the Examiner has failed to set forth a prima facie case of obviousness with respect to these Claims. For this additional reason, the Claims should be passed to issuance.[29]

---

[28] See Alexander Figure 17.
[29] In Re Oetiker, 977 F.2d 1443, 24 USPQ 2d 1443 (Fed. Cir. 1992).

II. Whether Claims 6 and 13-14 are unpatentable under 35 USC §103(a) over Alexander in view of U.S. Patent Number 5,996,064 to Zaidi et al. ("Zaidi") and further in view of U.S. Patent Number 6,345,362 to Bertin et al. ("Bertin").

Claim 6 depends from Claim 1. Similarly, Claims 13 and 14 depend from independent Claim 11. These Claims are patentable over the cited prior art for all of the reasons discussed above in reference to Claim 1.

In addition to the aspects discussed above in reference to Claim 1, Claims 6 and 13-14 further include the aspect of Applicants' invention wherein a count value is respectively associated with a particular instruction. As a result, each time the associated instruction is encountered in the instruction stream, the pipeline is partially de-piped to the level specified by the count value.[30] This capability allows certain known timing conflicts to be resolved in a very efficient manner.[31] This aspect of Applicants' invention is said to be taught by Zaidi.

Zaidi describes a mechanism for scheduling instructions within an out-of-order processor. The system inserts a "post-ready latency" delay between the execution of an identified instruction and the preceding instruction. Each instruction may be associated with a corresponding delay of this type[32]. Insertion of such delays ensures that proper operation occurs when the

---

[30] Specification page 29 line19 - page 30 line 2 in reference to Figures 8A, 8B, and 9.
[31] Specification page 8 line 27 - page 9 line 8.
[32] Zaidi column 7 lines 41 - 45.

processor is executing certain types of instructions.[33] This is summarized in Zaidi as follows:

> "The method includes receiving a plurality of instructions and determining the post-ready latency of each instruction. Thereafter, each instruction is scheduled for execution so that the instruction follows an earlier instruction by an amount of time at least equal to the post-ready latency of the instruction."[34]

As can be appreciated by the foregoing description, Zaidi provides a system much like that of Alexander. According to this system, delay is inserted between the times two successive instructions within the instruction stream become available to any of the execution units for execution.[35] Zaidi adds nothing to Alexander to teach or suggest Applicants' system and method for utilizing a specific count value to control the number of instructions that begins concurrent execution within the pipeline within a predetermined period of time. Moreover, with respective to the specific Claim limitations of Claims 6 and 13-14, Zaidi does not teach or suggest enabling a system such as Applicants' via recognition of a particular type of instruction. Finally, Zaidi does not teach using a count value associated with the particular instruction to control the number of instructions that begin execution within the pipeline in a predetermined amount of time. Thus, the addition of elements of Zaidi to the Alexander system does not provide any teaching or suggestion to render Applicants' Claims 6 and 13-14 obvious.

---

[33] Zaidi column 4 lines 47 - 62.
[34] Zaidi column 6 lines 1 - 6.
[35] Zaidi column 8 lines 64 - 67.

In addition to the foregoing, there is no motivation to add the Zaidi post-latency values to the Alexander system. Alexander discloses a system related to throttling instruction execution to reduce power consumption. This execution throttling is enabled based on temperature measurements. There is no reason in Alexander to recognize individual or specific instructions when utilizing this type of mechanism. Moreover, one skilled in the art would not be inclined to look to the Zaidi system, which describes a mechanism for controlling the execution of out-of-order instructions within a high-speed processor, for any teaching related to limiting power consumption within a processor. In fact, as discussed previously, the goal of providing optimal performance as set forth in Zaidi is generally at odds with the goal of providing power conservation as described in Alexander. Thus, there is no motivation, or benefit, in combining aspects of Alexander and Zaidi in the manner suggested by the Examiner. It has long been held that the mere fact that references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination.[36]

Next, the Examiner's assertions regarding Bertin are considered. The Examiner states that "Bertin demonstrated that it was known at the time of invention to attempt control of instructions based upon power usage."[37] As best understood, the Examiner appears to be asserting that the motivation to combine aspects of Zaidi with the Alexander system may be found in Bertin.

---

[36] *In re Mills,* 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990), emphasis added.
[37] Final Rejection, page 5, lines 7-8.

Bertin discloses a system to control the power consumption of various functional units, as may be useful within a portable electronic device. Functional units are in a high power state when instructions requiring their use are to be executed. Conversely, functional units are in a low power state when they are not involved in current instruction execution.[38] An execution unit allows execution of an instruction to continue at current processor speeds if the functional units that are required to execute the instruction are operating at the required power levels. Otherwise, the execution unit may stall the instruction stream for a time sufficient to allow voltage levels to ramp up to the required levels.[39]

The Bertin system appears to be particularly adapted for use with battery-powered portable integrated circuits.[40] This type of system is not adapted for use with a high-speed instruction pipeline. Specifically, the Bertin process of periodically powering on, then off, various functional units would not be feasible in modern high-speed processor pipelines that typically measure instruction cycles in nanoseconds or picoseconds. Such operations would decrease processing throughput beyond what would be practical. For this reason, one skilled in the art would not look to Bertin for any teaching concerning high-speed processor technology such as disclosed in Zaidi, or discussed in Applicants' Specification. Moreover, one would not be motivated by Bertin to include any aspects of the Zaidi out-of-order processor, which is designed to execute

[38] Bertin column 2 lines 28 - 33.
[39] Bertin column 3 lines 3 - 15.

instructions as fast as possible with minimal delay[41], with the Alexander system which throttles instruction execution to conserve power.

In addition to the foregoing, the Bertin system is based upon identifying instructions so that relevant functional units can be powered on and off as required by the identified instructions. The Examiner asserts that such functionality would be usefully incorporated into Alexander to identify energy hungry instructions.[42] This is not understood. As previously discussed, obtaining a measurable temperature change using a system such as Alexander will require modifying processor throughput a sizable amount that may involve hundreds, if not thousands, of instructions per second. Within this context, it is unclear how or why one would attempt to control such a throttling process based on the identification of individual instructions. No one instruction will provide any useful indication of power consumption levels, especially given the fact that modern instruction processors execute millions of instructions per second. Thus, rather than throttling processor throughput based on individual instruction identification, a workable approach employs temperature measurements as the enabling mechanism, as is taught by Alexander.[43]

For all of the foregoing reasons, one skilled in the art would not be motivated to combine any aspects of Bertin or Zaidi with Alexander. In sighting these unrelated teachings, the Examiner is attempting to piece together

---

[40] See, for example, Bertin column 1 lines 19 - 21.
[41] See, for example, Zaidi columns 2 - 6 discussing the goals of Zaidi, which involve achieving optimal performance in a digital processor.
[42] Final Rejection, page 5 lines 11 - 14.

Applicants' invention in hindsight, something that has long been held impermissible.[44]

To summarize, the combined references do not teach or suggest all aspect of Applicants' claimed invention with respect to Claims 6 and 13-14. Specifically, the references do not teach or suggest associating a particular instruction with a count such that, upon entry of this instruction into an instruction pipeline, a control system is enabled to allow a number of instructions as indicated by the count value to enter the pipeline within a predetermined period of time. Important benefits are provided by Applicants' invention. These benefits are not achieved by the prior art systems, alone or in combination. Moreover, the Examiner has failed to set forth a prima facie case of obviousness with respect to these Claims. There is no suggestion or motivation, either in the references themselves, or in the knowledge available to one of skill in the art, to modify the references in the manner cited by the Examiner. Claims 6 and 13-14 are therefore patentable over Alexander in view of Zaidi in further view of Bertin.

---

[43] Alexander column 11 line 14 et seq.

[44] *In re Gorman*, 933 F.2d 982, 986, 18 USPQ2d 1885, 1888 (Fed.Cir.1991).

Conclusion and Request for Relief

Alexander does not teach or suggest the invention claimed by Applicants' Claims 1, 2, 5, 11-12 and 19-20. Alexander teaches a prior art mechanism for inserting delay between instructions in the instruction stream. In contrast, Applicants' invention provides a more precise level of control that determines the number of instructions that will begin execution in a pipeline in a predetermined period of time. Furthermore, Zaidi and Bertin do not add anything to Alexander to teach or suggest this mechanism, or the additional aspects of Applicants' system described in dependent Claims 6, 13, and 14. The invention described in Claims 1-22 provides important benefits not realized by any of the cited references, alone, or in combination. For this reason, and because the Examiner has not met the requirements for a prima facie case of obviousness under 35 USC §103(a), Claims 1-22 are patentable over the cited references. It is therefore respectfully requested that the rejection of the Claims be overturned, and the Claims be passed to issue.

Respectfully submitted,

Beth L McMahon 07/01/04

Beth L. McMahon
Attorney for Applicants
Reg. No. 41,987
Telephone No. 651-635-7893
UNISYS Corporation
M.S. 4773
PO Box 64942
St. Paul, MN 55164-0942

Appendix

Presented is a clean set of Claims 1-22 as last amended August 2, 2002.

Claim 1:

1. For use in a data processing system having an instruction processor to execute instructions included in the instruction set of the instruction processor, the instruction processor having an instruction pipeline capable of initiating simultaneous execution on a variable number of the instructions in a predetermined period of time, a system for programmably controlling the variable number of the instructions beginning execution within the instruction pipeline during the predetermined period of time, comprising:

a first storage device to receive and to store a programmable count value indicative of a predetermined number of instructions; and

a logic sequencer coupled to said first storage device to receive said programmable count value, and in response thereto, to generate a pipeline control signal provided to the instruction pipeline to cause the instruction pipeline to receive, and to initiate concurrent execution on, the predetermined number of the instructions in the predetermined period of time.

Claim 2:

2. The system of Claim 1, and further including programmable enable logic to selectively enable said logic sequencer to be responsive to said programmable count value.

Claim 3:

3. The system of Claim 2, wherein said first storage device is coupled to receive a respective programmable count value for predetermined combinations of the instructions, and wherein said programmable enable logic includes circuits to selectively enable said logic sequencer to be responsive to said respective programmable count value when a respective one of said predetermined combinations of the instructions enters the instruction pipeline.

Claim 4:

4. The system of Claim 2, wherein said first storage device is coupled to receive a respective programmable count value for predetermined combinations of the instructions only if a predetermined condition occurs within the instruction pipeline, and wherein said programmable enable logic includes circuits to selectively enable said logic sequencer to be responsive to said respective programmable count value when a respective one of said predetermined combinations of the instructions enters the instruction pipeline and only if said predetermined condition occurs within the instruction pipeline.

Claim 5:

5. The system of Claim 1, and further including scan enable logic coupled to said logic sequencer to programmably enable said logic sequencer to repeatedly generate said pipeline control signal to initiate execution of said predetermined number of the instructions during successive periods of time that are each equal to the predetermined period of time.

Claim 6:

6. The system of Claim 2, wherein said first storage device is adapted to receive, and to store, a respective first one of said programmable count values for each of first selectable ones of the instructions, and wherein said programmable enable logic includes circuits to enable said logic sequencer to receive, for any of said first selectable ones of the instructions, said respective first one of said programmable count values when said any of said first selectable ones of the instructions enters the instruction pipeline to begin execution.

Claim 7:

7. The system of Claim 6, and further comprising:

a second storage device coupled to said first storage device to store each said respective first one of said programmable count values and to provided said each respective first one of said programmable count values to said first storage device as a respective one of said first selectable ones of the instructions enters the

instruction pipeline, said second storage device further to store respective first instruction combination signals for each of said first selectable ones of the instructions;

a third storage device coupled to said logic sequencer, said third storage device adapted to store respective second instruction combination signals for each of second selectable ones of the instructions; and

a compare circuit to enable said logic sequencer to be responsive to said respective first one of said programmable count values for an executing one of the instructions N+1 where said instruction N+1 is one of said first selectable ones of the instructions and if said respective first instruction combination signals for said instruction N+1 have a predetermined relationship to said respective second instruction combination signals for one of the instructions N wherein said instruction N is one of said second selectable ones of the instructions, and is further the instruction to enter the instruction pipeline immediately before said instruction N+1.

Claim 8:

8. The system of Claim 7, wherein said third storage device further includes circuits to store microcode instructions to control execution of extended ones of the instructions, predetermined ones of said microcode instructions being associated with associated ones of said respective second instruction combination signals; and

a microsequencer coupled to said third storage device to read out a respective sequence of said microcode instructions to control execution of a

respective one of said extended ones of the instructions that is being executed within the instruction pipeline, and whereby if any of said microcode instructions are said predetermined ones of said microcode instructions, to provided said associated ones of said respective second instruction combination signals to said compare circuit as said respective second instruction combination signals for said instruction N.

Claim 9:

9. The system of Claim 8, wherein said microsequencer includes conditional logic response to variable conditions within the instruction pipeline, and whereby and said respective sequence of said microcode instructions is read from said third storage device based on said variable conditions.

Claim 10:

10. The system of Claim 7, wherein said third storage device further includes circuits to store a respective second one of said programmable count values for each of said second selectable ones of the machine instructions, and further including a programmable selector coupled to said first storage device to programmably select between said respective second one of said programmable count values for said instruction N or said respective first one of said programmable count values for said instruction N+1 for use as said programmable count value.

Claim 11:

11. For use in an instruction pipeline of an instruction processor, the instruction processor to execute instructions that are part of the instruction set of the instruction processor, the instruction pipeline being adapted to initiate the execution of a variable number of instructions, up to a predetermined maximum number of instructions, within a predetermined period of time when the instruction pipeline is operating in a default mode, and whereby up to said predetermined maximum number of instructions may be executing simultaneously within the instruction pipeline, the instruction pipeline including a pipeline controller to generate a pipeline control signal for temporarily preventing ones of the instructions from entering the instruction pipeline, a method of utilizing the pipeline controller to control the number of instructions for which execution is initiated by the instruction pipeline within the predetermined period of time, comprising the steps:

providing a count to the pipeline controller; and

utilizing the pipeline controller to selectively assert the pipeline control signal to cause the instruction pipeline to initiate the execution of the number of instructions specified by said count within a period of time equal to the predetermined period of time.

Claim 12:

12. The method of Claim 11, wherein the pipeline controller includes a programmable enable circuit to selectively enable the generation of the pipeline control signal, and further including the step of:

programming the programmable enable circuit to enable the pipeline controller to repeatedly selectively assert the pipeline control signal such that the instruction pipeline initiates the execution of the number of instructions specified by said count during successive periods of time that are each equal to the predetermined period of time.

Claim 13:

13. The method of Claim 11, wherein the instruction processor includes a first memory device coupled to the pipeline controller, and further including the steps of:

storing within the first memory device respective first count signals for each of first predetermined ones of the instructions; and

providing said respective first count signals to the pipeline controller as said count when a respective one of said first predetermined ones of the instructions enters the instruction pipeline.

Claim 14:

14. The method of Claim 13, wherein the pipeline controller may be programmably enabled, and further including the step of:

enabling the pipeline controller to be responsive to said respective first count signals.

Claim 15:

15. The method of Claim 13, wherein the instruction processor includes a second memory device coupled to the pipeline controller, and further including the steps of:

storing within the first memory device respective first compare signals for each of said first predetermined ones of the instructions;

storing within the second memory device respective second compare signals for each of second predetermined ones of the instructions; and

comparing said respective first compare signals for an instruction N+1 that is one of said first predetermined ones of the instructions and that is executing within the instruction pipeline to said respective second compare signals for an instruction N that is one of said second predetermined ones of the instructions, and that entered the instruction pipeline for execution before said instruction N+1 entered the instruction pipeline, said comparing step performed to determine whether a predetermined relationship exists between said respective first compare signals for said instruction N+1 and said respective second compare signals for said instruction N;

and wherein said step of providing said respective first count signals to the pipeline controller is performed only if said predetermined relationship exists.

Claim 16:

16. The method of Claim 15, wherein the second memory device further stores microcode instructions to control the execution of ones of the instructions that are extended-mode instructions, and wherein the instruction processor includes a microsequencer to read a respective sequence of the microcode instructions from the second memory device to control execution of an instruction that is resident within the instruction pipeline, and further including the steps of:

associating predetermined ones of the microcode instructions each with a respective one of said respective second compare signals;

reading via the microsequencer the respective sequence of the microcode instructions from the second memory device for said instruction N when said instruction N is one of the extended mode instructions;

performing said comparing step using said respective second compare signals that have been associated with any said predetermined micro instruction included in said sequence of micro instructions.

Claim 17:

17. The method of Claim 16, wherein the microsequencer is responsive to variable conditions occurring within the instruction processor, and wherein said reading step is performed to select said respective sequence of the microcode instructions for said instruction N in response to said variable system conditions.

Claim 18:

18. The method of Claim 15, wherein the second memory device further stores respective second count signals for each of said second predetermined ones of the instructions, and wherein said step of providing said first respective count signals to the pipeline controller includes the step of selecting whether said respective second count signals will be substituted for use as said count instead of said first respective count signals.

Claim 19:

19. For use in an instruction processor having an instruction pipeline for executing multiple instructions concurrently, the instruction pipeline being capable of initiating concurrent execution for up to a predetermined maximum number of instructions within a predetermined period of time, a system for programmably controlling the number of instructions for which concurrent execution is initiated within the predetermined period of time, comprising:

storage means for receiving programmable count signals; and

sequencer means for responding to said programmable count signals, and for issuing a pipeline control signal to the instruction pipeline for controlling the entry of instructions into the instruction pipeline such that concurrent execution is initiated for the number of instructions specified by said programmable count signals within a period of time equal to the predetermined period of time.

Claim 20:

20. The system of Claim 19, wherein said storage means includes scan enable means for programmably enabling said sequencer means to issue said pipeline control signals continually such that concurrent execution is initiated for the number of instructions specified by said programmable count signals within successive periods of time each equal to the predetermined period of time.

Claim 21:

21. The system of Claim 20, and further including instruction combination means for providing respective ones of said programmable count signals to said storage means when an associated predetermined combination of the instructions has entered the instruction pipeline.

Claim 22:

22. The system of Claim 21, wherein said instruction combination means includes means for responding to variable conditions occurring within the instruction processor, and whereby said instruction combination means provides respective ones of said programmable count signals to said storage means after said associated predetermined combination of instructions has entered the instruction pipeline, and following the occurrence of a predetermined one of said variable conditions.